THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Inference of Effective Pairwise Relations for Data Processing

FAZELEH SADAT HOSEINI

Division of Data Science and AI
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2020

Inference of Effective Pairwise Relations for Data Processing

FAZELEH SADAT HOSEINI

ISSN 1652-876X
Department of Computer Science & Engineering
Division of Data Science and AI
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2020.

"Never give up on a dream just because of the time it will take to accomplish it. The time will pass anyway"
- Earl Nightingale

Abstract

In various data science and artificial intelligence areas, representation learning is a performance-critical step. While different representation learning methods can detect different descriptive and latent features, many representation learning methods reflect on pairwise relations. The thesis consists of two parts, studying pairwise relations from two points of view: i) Pairwise relations between the states of a Markov chain. ii) Pairwise relations between objects in a dataset based on a desired (dis)similarity measure.

In the first part of the thesis we consider Markov chains, noting that pairwise relations between its states are naturally modelled by the state-transition matrix. We propose a method for modeling the performance of a synchronization method for a multi-processor architecture. Our model introduces and builds upon a cache line bouncing process that models the interaction of threads accessing the shared cache lines.

In the second part of the thesis, we consider representation learning using the transitive-aware Minimax distance, which enables the extraction of elongated manifolds and structures in the data. While recent work has made Minimax distances computationally feasible, little attention has been put to its memory footprint, which is naturally $\mathcal{O}(N^2)$, the cost of storing all pairwise distances. We do however compute a novel hierarchical representation of the data, requiring $\mathcal{O}(N)$ memory, from which pairwise Minimax distances can then be efficiently inferred, in total requiring $\mathcal{O}(N)$ memory, at the cost of higher computational cost.

An alternative sampling-based approach is also derived, which computes approximate Minimax distances, also in $\mathcal{O}(N)$ memory but with a significantly reduced computational cost, while still yielding a good approximation, as verified by impressive results on a clustering benchmarks.

Finally, we develop an unsupervised learning framework for clustering vehicle trajectories based on Minimax distances. The performance of the framework is validated on real-world datasets collected from real driving scenarios, on which satisfactory performance is demonstrated.

Keywords: Representation Learning, Pairwise Relations, Minimax Distance, Memory Efficiency, Motion trajectory clustering, Concurrent programming, Performance Modeling

Acknowledgment

I am most grateful to my main supervisor Morteza Haghir Chehreghani for the invaluable guidance and support I receive from him. I am also grateful to Alexander Schliep, Philippas Tsigas, Vincenzo Gulisano, and Marina Papatriantafilou for many insightful comments and help from them, as well as my examiner Peter Damaschke.

My journey at Chalmers would not have been the same without all of my colleagues. Many thanks to Adones, Ali, Aljoscha, Amir, Aras, Arman, Ashkan, Bastian, Bei, Beshr, Birgit, Boel, Carlo, Charalampos, Christos, Dag, Debabrota, Devdatt, Dimitris, Divya, Elad, Elena, Emil, Emilio, Francisco, Fredrik, Georgia, Graham, Hampus, Hannah, Iosif, Ivan, Joris, Karl, Katerina, Magnus, Marika, Marwa, Mehrzad, Nasser, Niklas, Olaf, Oliver, Richard, Romaric, Shirin, Simon, Thomas, Tobias, Tomas, Valentin, Vladimir and Wissam. I would also like to thank the administrative staff, especially Eva, Clara, Marianne, and Rebecca.

I would like to express my deepest gratitude to my parents for their endless love and countless sacrifices to guarantee that I would be able to follow the path I wanted. To my sister and brother, who always support me. To my wonderful friends for providing me pleasant distractions. And, of course, to my incredible partner, Lucas, for his love and belief in me.

This research has been funded by Swedish Research Council, project ”Models and Techniques for Energy-Efficient Concurrent Data Access Designs” grant nr. 201605360, as well as faculty funds, IKB 37403230, KST 372300.

Fazeleh Sadat Hosini
Göteborg, December 2020

List of Publications

Appended publications

This thesis is based on the following publications:

I Fazeleh Hoseini, Aras Atalar, Philippas Tsigas
" Modeling the Performance of Atomic Primitives on Modern Architectures"
Appeared in the *48th International Conference on Parallel Processing (ICPP), 28:1-11, 2019.*

II Fazeleh Hoseini, Morteza Haghir Chehreghani
" Memory-Efficient Minimax Distance Measures"
Technical report arXiv:2005.12627
Submitted to *Machine Learning Journal: Special Issue on Foundations of Data Science 2021.*

III Fazeleh Hoseini, Sadegh Rahrovani, Morteza Haghir Chehreghani
"A Generic Framework for Clustering Vehicle Motion Trajectories"
Technical report arXiv:2009.12443
Under submission, 2021.

Contents

Chapter 1

Introduction

The difficulty of many data processing tasks can vary immensely depending on how the data representation is inferred, thus a proper data representation can be crucial. To this end, representation learning is concerned with the development of machine learning algorithms to discover useful representations and latent features, and consequently is the first step in many machine learning and data analytic tasks.

Representation learning can be categorized as traditional feature learning and deep learning models [1]. Traditional feature learning is a broad family consisting of algorithms that can be supervised or unsupervised, linear or nonlinear, generative or discriminative, global or local. In general, global methods try to extract global information from data in the learned feature space, while local methods focus on preserving local similarity between data when learning the new representations.

Some representation learning algorithms take pairwise relations or graph structure into consideration. The pairwise relation between data points, mostly defined by similarities or dissimilarities, is an essential concept for many machine learning algorithms such as Kernel PCA [2], discriminant analysis (LDA) [3], isomap [4], kernel methods [5], and t-SNE [6].

Pairwise relations can be useful not only for a static set of data points, but also for modeling dynamic processes. Markov chains are statistical models that describe a sequence of events, where the probability of each event occurring depends on the state of the previous event. A Markov chain is determined by its state transition matrix, which defines the

probability of a transition from each of the states to the others. Now, for applications that will be covered in this thesis, pairwise relations between the states themselves can be useful for modeling the state transition probabilities.

In several applications, the underlying structures and patterns in data are better represented when extracting transitive relations, rather than using the direct (dis)similarities. The meaning of a transitive relation is that it is inferred from different paths that connect the data points with each other. In particular, for the clustering application the Minimax distance between a pair of data points, i.e. the minimum largest gap among all possible paths between them, exploits the local geometry of data points while extracting transitive relations between them in such a way that elongated manifolds and structures can easily be separated.

In this thesis, an important concept for modeling pairwise relations is the Minimax distance. The Minimax distance is an unsupervised and non-parametric representation learning technique, for which prior knowledge about the shape of the clusters and the structure of data is not required. Although the main drawback of the Minimax distance technique is that for a standard algorithm the computational and memory cost is high, there have been recent studies to improve the computational aspect [7]. In contrast, we focus on reducing the memory cost, for which, to the best of our knowledge, there has been no prior research.

The contributions of this thesis, which are carried by three papers, are as follows:

- **Paper I:** Introduces a method to model the performance of a synchronization method on multi-processors by considering the core's pairwise relations.

- **Paper II:** Develops two approaches for the memory-efficient computation of Minimax distances requiring a linear memory with respect to the size of the dataset.

- **Paper III:** Develops an unsupervised framework based on Minimax distances to cluster vehicle trajectories. This framework does not require fixing hyper-parameters and can be applied as a validation tool to assess the quality of synthetic trajectories.

The rest of the thesis is organized as follows. Chapter 2 provides background on representation learning and concurrent programming. Chapter 3 describes the research challenges covered by each paper and the contributions to address these challenges. Finally, Chapter 4 concludes the thesis and discusses future work.

Chapter 2

Background

In this chapter, we cover some background information on representation learning. Also, we present preliminaries to two case studies: concurrent programming and vehicle motion trajectories.

2.1 Representation Learning

Quite generally, data can be represented by a set of objects characterized by either feature vectors or pairwise dissimilarities. In particular, object (dis)similarities can yield valuable knowledge and can be helpful for a lot of applications in different machine learning domains [8], such as semantic similarity [9], recommendation systems [10], and object detection [11] in machine vision.

Furthermore, a graph-based representation of object (dis)similarities provides not only a simple way to picture object relationships, but can also be used to represent (dis)similarities numerically, in a straightforward manner. Consider a dataset $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N\}$ where $\mathbf{d}_i$ is the feature vector of object $i \in \mathbf{O}$ and $\mathbf{O} = \{1, 2, \ldots, N\}$ is the corresponding index set. $\mathbf{D}$ can be represented by a graph $G(\mathbf{O}, \mathbf{E})$ where the nodes are denoted by the object indices $\mathbf{O}$, and the edge weights are determined by a dissimilarity function, e.g. the squared Euclidean distance between object features defined by $\mathbf{E}_{ij} = |\mathbf{d}_i - \mathbf{d}_j|^2,\ i, j \in \mathbf{O}$.

A graph-based data representation allows us to consider *link-based* distances between the objects. Link-based distances are unsupervised

representations and distance measures such that for a pair of nodes in a graph, all the *paths* between them are considered. Based on this general definition, we can distinguish three categories of link-based distances: i) Distance measures that are defined as the summation of edge weights along a path connecting a pair of objects [12]. ii) Distance measures that are based on the shortest path between a pair of objects [13]. iii) Minimax distance measures, which are determined by the minimum largest gap along all possible paths between a pair of objects. Link-based distances are useful for detecting transitive relations in data; thus, these measures are suitable for separating data that is clustered into non-convex manifolds [14].

2.1.1 The Minimax Distance

The Minimax distance is an efficient link-based measure which determines the minimum largest gap among all feasible paths between the objects in a graph. Minimax distances perform better than other link-based methods in particular to extract the transitive relations [15]. Consider again a graph $G(\mathbf{O},\mathbf{E})$. Next, consider the following property, which is highly desired for our applications, and strictly fulfilled for the Minimax distance:
If object $i \in \mathbf{O}$ is similar to $l \in \mathbf{O}$, and l is similar to $j \in \mathbf{O}$, then objects i and j are considered as similar objects as well, even if their direct pairwise similarity is low.
Due to the above property, we can conclude that the Minimax distance is a transitive-aware distance measure.

Minimax distances have been first studied on path-based clustering applications [16], and more recently in K-nearest neighbor search [12, 14, 17]. The classic approach for obtaining pairwise distances of all pairs of objects within a dataset is to use a modified variant of Floyd-Marshall algorithm with a computational cost of $\mathcal{O}(N^3)$ [18], where N is the size of the dataset.

A more efficient algorithm to obtain pairwise Minimax distances proposed in [7] requires only $\mathcal{O}(N^2)$ runtime. The approach is based on the fact that Minimax distances over a graph are preserved for a Minimum Spanning Tree (MST) of the graph [19], and can thus be computed for the MST instead, which is more efficient. For an undi-

rected connected graph $G(\mathbf{O}, \mathbf{E})$, a spanning tree is a connected acyclic sub-graph of G which covers all vertices of G. A graph G might have more than one spanning tree. If G has positive edge weights, a minimum spanning tree for G is a tree that gives the minimum total cost (sum of edge weights). Furthermore, if all edge weights in G are identical, then the minimum spanning tree T over G is unique; otherwise, there are more than one MST with equal cost.

Assuming we are interested to obtain Minimax distances between all pairs of N objects in our dataset, we need to consider all edges. That is, we are dealing with a complete graph, for which the MST is to be computed. Prim's and Kruskal are two well-known algorithms to obtain an MST with a runtime of $\mathscr{O}(|\mathbf{E}| + |\mathbf{O}|log|\mathbf{O}|)$ and $\mathscr{O}(|\mathbf{E}|log|\mathbf{O}|)$ respectively, making Prim's preferable for complete graphs. Furthermore, for complete graphs, the memory requirement is $\mathscr{O}(N^2)$ for both algorithms.

In order to compute pairwise Minimax distances of a dataset efficiently, we can first form a minimum spanning tree $T(\mathbf{O}, \mathbf{E}_T)$, where $\mathbf{E}_T \in \mathbf{E}$. Since T is a tree, there is a unique path between any two nodes, i, and j. Thus, the longest edge along this path is the Minimax distance between i and j.

2.2 Concurrent Programming

Multi-processor architectures provide the possibility of parallel computation. A multiprocessor contains two or more processing units, cores, which can execute threads simultaneously. These cores might share caches at some level. Parallel threads executing on a multi-processor chip interact through shared memory. In an asynchronous thread execution, threads can execute at a different pace; thus, the steps of the threads can be interleaved. To maintain the correctness of these threads, we need to implement the appropriate synchronization.

2.2.1 Atomic Primitives

Atomic primitives (atomics) are hardware instructions that consist of a number steps to be executed atomically (without interruption). Atom-

ics ensure consistency and correctness in the presence of concurrent accesses. The most common application of atomics is when multiple threads are modifying a shared object. Since each thread's modification needs to be serialized to ensure correctness, atomics can be seen as a congestion point that can limit the performance and scalability of multi-threaded programs.

Atomics are massively used in designing concurrent data structures [20–22]. A critical property of atomics is the consensus number, which indicates the highest number of threads reaching a consensus with the atomic primitive in a wait-free fashion [20].

In this thesis, the most common atomic primitives are studied: Compare And Swap (CAS), Fetch And Increment (FAI), Test And Set (TAS), which are shown in Algorithm 1a, Algorithm 1b, and Algorithm 1c, with consensus number ∞, 2, and 2, respectively [20].

Algorithm 1a Compare And Swap (CAS) function

```
1: function CAS(int* p, int old, int new)
2:     if *p = old then
3:         Return False
4:     end if
5:     *p ← new
6:     Return True
7: end function
```

Algorithm 1b Test And Set (TAS) function

```
1: function TAS(int* p)
2:     int old ←* p
3:     *p ← True
4:     Return old
5: end function
```

Algorithm 1c Fetch And Increment (FAI) function

```
1: function TAS(int* p, int add)
2:     int old ←* p
3:     *p += add
4:     Return old
5: end function
```

2.2.2 Cache Coherency Protocol

Each core in a multi-processor system has its private cache, called L1 cache. A core might have more than one exclusive cache. When a core executes a thread on a specific memory location, the targeted memory location should be available in L1 of the core. However, there might be some copies of the target memory address in other caches. To guarantee the consistency of shared data resources that are locally available in multiple caches, we need cache coherency protocols. Cache coherency protocols are classified according to the number of cache states. In this thesis we focus on MESI [23] and MESIF [24], explained below.

The MESI protocol consists of four states, Modified (M), Exclusive (E), Shared (S), Invalid (I). In MESIF, we have the fifth state called Forward (F). In the Modified state, the content is exclusive to the current cache and has been modified from the main memory value. In the Exclusive state, the content is exclusively in the current cache but is the same as the main memory value. The Shared state indicates that other caches have an identical copy of this cache line, and it is the same as the main memory value. The Invalid state means that the value in the current cache is invalid. In MESIF, we have one more state, called Forward. If there are multiple copies of a memory address in different caches, one of them holds the F state to forward the cache value in the response of a copy request.

2.2.3 Execution Time

The execution time of atomic primitives are determined by two factors, which we shall return to after explaining the concept of thread contention.

Thread contention is a state in which one or more threads are waiting to access and modify a cache line. In the event of a contention, one thread gets the exclusive ownership of the cache line and executes atomics while others have to stall and try to win exclusive ownership after the current atomic execution terminates. Once the current execution terminates, based on the scheduling policy of the hardware, another thread acquires the cache line. A period that a thread needs to stall between requesting to execute an atomic on a specific cache line and

acquiring the cache line defined as stall time, and it is the first factor of the execution time. Therefore, the contention affects the stall time for an atomic primitive.

The second factor is the atomic execution time, which is the time between acquiring the cache line and finishing the atomic execution . This factor depends on the system's state when the thread acquires the cache line. For example, if the specified cache line is already in L1 of the core running the thread, the execution takes less time than the case that the cache line should be brought to the L1 of the core from memory.

2.3 Vehicle Motion Trajectories

In various real-world applications, there are vast amounts of raw data that can help to improve the application. One such application is Autonomous Drive (AD) vehicles, which is highly data-dependent, but for which vast amounts of raw data can be collected by sensors. Possible AD use cases include vehicle software development as well as verification.

However, although raw data may be available in abundance, further processing of the data can be very valuable for the task at hand. As letting human experts carry out such processing is very costly, a representation learning approach is highly preferable.

More specifically, in this thesis we consider unsupervised representation learning on a database of driving scenarios (vehicle motion trajectories), for two intended AD use cases: i) Automatic driving scenario specification, for vehicle software verification purposes. ii) Live driving scenario detection by an AD. In both cases, we do not assume any specification on the data, but yet identify a common value in extracting patterns and structures in the data.

We will now give further details on the motivation for the first use case, i.e. automatic driving scenario specification. In order to confidently assess the safety of AD vehicles through test driving in-the-wild, statistical analyses have shown that more than hundreds of millions of kilometers would have to be driven. One step in the direction to alleviate these issues is known as scenario-based verification, where one creates a scenario database by collecting data, from which driving

scenarios are then extracted. If the scenarios represent and cover the real-world scenarios well, such a database can be used as a source for generating test cases for AD verification, in virtual as well as real test driving environments. We assume that raw data of vehicle motion trajectories is available, and consider clustering such trajectories into driving scenarios, in an unsupervised manner. We regard the motion trajectories to be centered around an ego vehicle, and to be constituted by a sequence of the relative position of a nearby vehicle, the behavior of which is to be classified into a driving scenario. Trajectories can vary in length, which introduces some challenge. The driving scenarios (unknown by the algorithm) could be e.g. a cut-in by an overtaking vehicle.

There are different approaches to extract driving scenarios from raw data collected by sensors [25], one of which is clustering. A clustering approach to the problem aims at grouping the available data into meaningful driving scenarios. Previous work on trajectory clustering include K-means trajectories clustering [26], which however suffers from local optima and inability to extract elongated clusters. Clustering based on Hidden Markov Model [27] and Neural Network models [28] have also been pursued, but at great computational expense, yielding unsuitable methods for real-time application. Moreover, they might need labeled data for training, as well as they are very sensitive to some certain parameter settings.

As discussed, availability of a large and diverse scenario database is vital for the development and verification of AD vehicles. However, the amount of diverse real data available can rarely be too much, and class imbalance can often be expected. Consequently, rather than verifying AD solutions only on a limited amount of real data collected from the field, augmentation with synthetically generated, but realistic, motion trajectories is desirable. To this end, one approach is to use Generative Adversarial Nets [29], which combine a generative model that tries to capture the data distribution with a discriminative model that predicts whether data is synthetically generated by the generative model or whether it is a real data sample.

Chapter 3

Challenges and Contributions

3.1 Paper I

There are many studies and benchmarks on atomic primitives in the literature. However, these studies are proposed under full control over the state of the system, including cache coherency state, the location of copies of the cache line, exact shared memory location, and interleaving threads accessing the shared memory locations. It means that for measuring an element of the performance, for example, the latency of accessing a cache line with a particular cache location and coherency state, the benchmark should first bring the cache line to the modified state in the L1 cache by executing a thread modifying a shared memory location. It then signals another thread that is pending to access the shared memory location and measure the latency. Nevertheless, in the real world, there is no control over the state of the system.

In Paper I, we tackle this research gap by focusing on the pairwise relation of cache lines in the cores while executing atomics. Our method relies on modeling cache line bounces between threads accessing the shared cache lines. In our approach, by observing a sequence of threads executing the atomic, we conduct the sequence of cores accessing the shared memory. Then, we model this sequence of events with a stochastic process that maintains the Markov property. We represent the process with a Markov Chain and model the state transition matrix based on the hardware characteristics, core count, and proximity. Finally, having the state transition matrix and cost of each transition, we model the perfor-

mance. Our study defines performance in terms of latency, throughput, fairness, and energy consumption of atomic primitives.

3.2 Paper II

Computing pairwise relations, including Minimax distance, is costly in terms of computation and memory. Although there are several studies on computationally efficient methods to compute pairwise Minimax distances [7], memory-efficient methods for pairwise Minimax distance have not received proper attention in the literature. To the best of our knowledge, no earlier work considers the memory-efficient Minimax distance there has not been any improvement to the memory requirement of $\mathcal{O}(N^2)$ for this problem.

In Paper II, we propose two approaches for memory-efficient Minimax computation with linear memory requirement. The first approach is a novel algorithm to obtain, encode, and store pairwise Minimax distances of N objects with a linear memory requirement $\mathcal{O}(N)$. Also, we introduce incremental Prim's algorithm, which allows us to form a minimum spanning tree over a dataset without requiring the corresponding graph. Despite the memory-efficiency of this algorithm, its computational cost is high. Therefore we propose a second approach, which is a generic framework for efficient sampling-based on Minimax distances. The obtained sample set by this method has maximum consistency with the Minimax distances over all objects.

3.3 Paper III

Using collected raw sensor data, to label and extract driving scenarios of interest can be done via different approaches. Knowledge-based approaches enable us to extract scenarios based on prior fixed rules and defining a scenario description with some threshold that needs to be set in advance. The main advantages with this approach is having insight and control on all steps, and the use of prior knowledge about driving scenarios. However, similar to many rule-based systems, it can be subject to errors such as bias and missing unknown cases. Therefore, as a complementary approach we investigate using exploratory data

management and machine learning tools, in particular clustering, to accelerate and verify the obtained scenario labels. The main benefit of this approach is to find some unknown scenarios, patterns or outlier trajectories, especially since the rule based tools depend on the scenario threshold values and thus they might miss or misclassify those events.

In Paper III, we focus on developing an unsupervised learning framework that enables us to cluster given vehicle motion trajectories to different driving scenarios. Our approach is based on Minimax distance and does not require tuning critical hyper-parameters. Also, our method can successfully handle trajectories of varying lengths. Finally, we utilize our framework to validate the quality of synthetically generated driving trajectories, comparing with the original driving scenarios.

Chapter 4

Conclusion and Future Work

In this thesis, we have studied pairwise relations in data processing from two perspectives. On the one hand, we considered Markov chains, noting that pairwise relations between its states are naturally modelled by the state-transition matrix. Taking on this perspective, we proposed a method for modeling the performance of a synchronization method for a multi-processor architecture. Several performance metrics were covered, including energy consumption, throughput, latency, and fairness. Our model is based on a cache line bouncing process which models interacting threads, accessing shared cache lines.

On the other hand, we considered representation learning using the transitive-aware Minimax distance, enabling us to reveal elongated manifolds and structures in the data. We focused our attention to the memory-efficiency of Minimax distance algorithms, and discovered that little work had been done in that regard, unlike for computational efficiency. While the memory footprint of a Minimax algorithm is naturally that of storing all pairwise distances, $\mathcal{O}(N^2)$, we propose a novel hierarchical representation of the data, requiring only $\mathcal{O}(N)$ memory, and from which pairwise Minimax distances can be efficiently inferred. Our algorithm encodes the data into the hierarchical representation, and can compute exact Minimax distances entirely in $\mathcal{O}(N)$ memory, at the expense of a higher computational cost.

Next, an alternative sampling-based approximate approach is proposed. While this algorithm also works in linear memory, the computational load is significantly reduced. The resulting approximate

Minimax distance are very adequate, as verified by satisfactory results on a clustering benchmarks.

Moreover, we proposed an unsupervised framework to cluster vehicle motion trajectories based on Minimax distances. Our framework does not demand setting hyper-parameters. Also, it can be used to validate synthetic data generated by Generative Adversarial Networks (GANs). We perform evaluations on real-world datasets collected from real driving scenarios, as well as synthetic datasets generated by GAN, demonstrating impressive performance of the framework.

In future work, we will explore how to apply Minimax distances in a noisy environment since Minimax distance, as considered so far, is not robust to noise. Furthermore, instead of assuming that the base pairwise relations of objects are given (explicitly or implicitly), in the next step, we will combine Minimax distances with online learning, while learning pairwise distances incrementally over time.

Bibliography

[1] G. Zhong, L.-N. Wang, X. Ling, and J. Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," *The Journal of Finance and Data Science*, vol. 2, no. 4, pp. 265–278, 2016.

[2] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel pca and de-noising in feature spaces," *Advances in neural information processing systems*, vol. 11, pp. 536–542, 1998.

[3] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, "Fisher discriminant analysis with kernels," in *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*. Ieee, 1999, pp. 41–48.

[4] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford, "The isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, pp. 7–7, 2002.

[5] B. Schölkopf, "The kernel trick for distances," *Advances in neural information processing systems*, vol. 13, pp. 301–307, 2000.

[6] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[7] M. H. Chehreghani, "Efficient computation of pairwise minimax distance measures," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 799–804.

[8] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[9] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *arXiv preprint cmp-lg/9709008*, 1997.

[10] G. Arora, A. Kumar, G. S. Devre, and A. Ghumare, "Movie recommendation system based on users' similarity," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 4, pp. 765–770, 2014.

[11] T. Ma and L. J. Latecki, "From partial shape matching through local deformation to robust global shape similarity for object detection," in *CVPR 2011*. IEEE, 2011, pp. 1441–1448.

[12] K.-H. Kim and S. Choi, "Neighbor search with global geometry: a minimax message passing algorithm," in *Proceedings of the 24th international conference on machine learning*, 2007, pp. 401–408.

[13] F. Buckley and F. Harary, *Distance in graphs*. Addison-Wesley, 1990.

[14] K.-H. Kim and S. Choi, "Walking on minimax paths for k-nn search." in *AAAI*, 2013.

[15] S. Le Moan and C. Cariou, "Minimax bridgeness-based clustering for hyperspectral data," *Remote Sensing*, vol. 12, no. 7, p. 1162, 2020.

[16] B. Fischer and J. M. Buhmann, "Path-based clustering for grouping of smooth curves and texture segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 513–518, 2003.

[17] M. H. Chehreghani, "K-nearest neighbor search and outlier detection via minimax distances," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 405–413.

[18] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[19] R. B. Zadeh and S. Ben-David, "A uniqueness theorem for clustering," *arXiv preprint arXiv:1205.2600*, 2012.

[20] M. Herlihy, N. Shavit, V. Luchangco, and M. Spear, *The art of multiprocessor programming*. Newnes, 2020.

[21] B. Norris and B. Demsky, "Cdschecker: checking concurrent data structures written with c/c++ atomics," in *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages and applications*, 2013, pp. 131–150.

[22] M. M. Michael and M. L. Scott, "Simple, fast, and practical non-blocking and blocking concurrent queue algorithms," in *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, 1996, pp. 267–275.

[23] G. F. Pfister, *In search of clusters*. Prentice Hall PTR Upper Saddle River, NJ, 1998, vol. 2.

[24] R. Intel, "Intel 64 and ia-32 architectures optimization reference manual," *Intel Corporation, Sept*, 2014.

[25] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1821–1827.

[26] Q.-m. PAN, Z. Wen-hui, and Y.-m. CHENG, "Trajectory classification and recognition of moving objects," *Fire Control and Command Control*, vol. 11, 2009.

[27] F. Porikli, "Trajectory distance metric using hidden markov model based representation," in *IEEE European Conference on Computer Vision, PETS Workshop*, vol. 3, no. 2004. Citeseer, 2004.

[28] W. Hu, D. Xie, T. Tan, and S. Maybank, "Learning activity patterns using fuzzy self-organizing neural network," *IEEE Transactions*

on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 34, no. 3, pp. 1618–1626, 2004.

[29] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.